# MKNOD

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-02

## Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 7541 bytes

| | |
|---|---|
| **Attack Category** | • Path spoofing or confusion problem |
| **Vulnerability Category** | • Indeterminate File/Path<br>• TOCTOU - Time of Check, Time of Use |
| **Software Context** | • File Creation |
| **Location** | • sys/stat.h |
| **Description** | The mknod function creates a new file (or directory or special file) called pathname with theMode as the mode. The file type and permissions of the new file are initialized from mode. mknod() is often used to create device files.<br><br>mknod() is vulnerable to TOCTOU attacks.<br><br>A call to mknod() should be flagged if the first argument (the file name) is used previously in a check-category call. |
| **APIs** | **Function Name**: mknod — **Comments**: use |
| **Method of Attack** | The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.<br><br>The mknod() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability.<br><br>A TOCTOU attack in regards to mknod() can occur, for example, when<br><br>a. A check for the existence of a filename (check call) occurs |

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

| | |
|---|---|
| | b. mknod() is executed<br><br>Between a and b, an attacker could, for example, link the target directory/file (the one to be opened) to a different known directory or file.The subsequent mknod() call would either fail or have unexpected results or behavior. |
| **Exception Criteria** | |
| **Solutions** | |

| **Solution Applicability** | **Solution Description** | **Solution Efficacy** |
|---|---|---|
| Generally applicable to all mknod() calls. | Utilize a file descriptor version of check and use functions. | Effective. |
| Generally applicable to all mknod() calls. | The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity can not be assured, but it does help to limit the false sense of security given by the check. | Does not resolve the underlying vulnerability but limits the false sense of security given by the check. |
| Generally applicable to all mknod() calls. | Limit the interleaving of operations on files from multiple processes. | Does not eliminate the underlying vulnerability but can help make it more difficult to exploit. |
| Generally applicable to all mknod() calls. | Limit the spread of time (cycles) between the "check" and "use" of a resource. | Does not eliminate the underlying vulnerability but can help make it more |

| | | |
|---|---|---|
| | | difficult to exploit. |
| Generally applicable to all mknod() calls. | Recheck the resource after the use call to verify that the action was taken appropriately. | Effective in some cases. |

| | |
|---|---|
| **Signature Details** | int mknod (const char *path , mode_t mode, dev_t dev) |
| **Examples of Incorrect Code** | `/* Same as positive, except a check call has been added */`<br><br>`#include "sys/types.h"`<br>`#include "sys/stat.h"`<br><br>`dev_t dev;`<br>`int status;`<br>`int check_status;`<br>`struct stat statbuf;`<br>`...`<br>`check_status=stat("/home/cnd/mod_done", &statbuf);`<br><br>`status = mknod("/home/cnd/mod_done", S_IFIFO \| S_IWUSR \| S_IRUSR \| S_IRGRP \| S_IROTH, dev);` |
| **Examples of Corrected Code** | `/* The following example shows how to create a FIFO special file named /home/cnd/mod_done, with */`<br>`/* read/write permissions for owner, and with read permissions for group and others. */`<br>`/* This would be considered a 'better' example because no check call exists. Given however that */`<br>`/* mknod is often used to create device files, so check calls may be common. This class of solution then */`<br>`/* may not be very appropriate. */`<br><br>`#include "sys/types.h"`<br>`#include "sys/stat.h"`<br><br>`dev_t dev;`<br>`int status;`<br>`...`<br>`status = mknod("/home/cnd/mod_done", S_IFIFO \| S_IWUSR \|` |

|  |  |
|---|---|
|  | `S_IRUSR \| S_IRGRP \| S_IROTH, dev);` |
| **Source References** | • Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, pg. 222<br>• UNIX man page for mknod()<br>• The IEEE and The Open Group. "mknod - make a directory, a special file, or a regular file[2]." The Open Group Base Specifications Issue 6; IEEE Std 1003.1, 2004 Edition (2004). |
| **Recommended Resource** |  |
| **Discriminant Set** | **Operating System**: • UNIX<br>**Languages**: • C • C++ |

## Cigital, Inc. Copyright

1. mailto:copyright@cigital.com